SedNet™ 400 Mbps IEEE-1394 VME Communication Adapter

with optional IP module carrier board

User's Manual

Edition 1

SEDERTA

Product Division of Mindready Solutions

Sederta, Product Division of Mindready Solutions Inc. License Agreement

IMPORTANT – READ CAREFULLY BEFORE INSERTING THE INSTALLATION CD-ROM INTO A CD-ROM DRIVE. By inserting the installation CD-ROM in a CD-ROM drive, you indicate acceptance of the following Sederta, Product Division of Mindready Solutions Inc. License Agreement.

This is a legal agreement between you (either an entity or an individual) and Sederta, Product Division of Mindready Solutions Inc. By inserting the installation CD-ROM in a CD-ROM drive you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unused software and accompanying items (including written materials) to the place of origin from which you obtained them for a full refund.

1. GRANT OF LICENSE. Sederta, Product Division of Mindready Solutions Inc. grants you the right to use the enclosed Sederta, Product Division of Mindready Solutions Inc. software files (the "SOFTWARE") on any computer equipped with a Sederta SedNet adapter card. Use of the SOFTWARE with non-Sederta hardware is not permitted.
2. COPYRIGHT. The SOFTWARE is owned by Sederta, Product Division of Mindready Solutions Inc. and is protected by Canadian copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g. musical recording or book) except that you may make one copy of the SOFTWARE solely for backup purposes. You may not copy the written materials accompanying the SOFTWARE.
3. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and all accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. If the SOFTWARE is an update or has been updated, any transfer must include the most recent update and all prior versions. You may not reverse engineer, de-compile, or disassemble the SOFTWARE.

This Agreement is governed by the laws of Canada.

LIMITED WARRANTY

Sederta, Product Division of Mindready Solutions Inc. reserves the right to make changes to its products or services without notice or obligation to notify any person of such changes. Sederta, Product Division of Mindready Solutions Inc. is not responsible for any damages caused by using this product. All technical information and recommendations in this guide are believed reliable, but the accuracy and completeness thereof are not guaranteed or warranted, and they are not intended to be, nor they should be understood to be.

SedNet is a trademark of Sederta Inc. All other product names mentioned herein are the trademarks of their respective owners.

Disclaimer

Please read and abide by the following disclaimer carefully. Any questions or comments you may have should be directed to:

Technical Publications Department
Sederta, Product Division of Mindready Solutions Inc.
1375 Trans-Canada Highway, Suite 400
Dorval, Quebec
Canada
H9P 2W8

Sederta, Product Division of Mindready Solutions Inc. makes no warranty, whether expressed or implied, with respect to this documentation manual, including any warranties of merchantability or fitness for a particular purpose. Sederta, Product Division of Mindready Solutions Inc. is not responsible for erroneous material that this document may contain. Furthermore, all information in this document is subject to change at any time without prior notice.

Table of Contents

List of Tables

List of Figures

How to Reach Us

Sederta, Product Division of Mindready Solutions Inc.
1375 Trans-Canada Highway, Suite 400
Dorval, Quebec
Canada
H9P 2W8

For information, comments or suggestions, please contact us by E-mail at info@sederta.com

There are several ways to reach us for technical support:

Tel.: +1 (514) 338-8749
Fax: +1 (514) 338-1915
E-mail: support@sederta.com

Sederta, Product Division of Mindready Solutions Inc. also offers a wide range of services to support our customers' needs throughout their projects:

- Consulting Services
- Customizable Product Support Programs
- On-Site Assistance
- Comprehensive Technical Assistance
- IEEE-1394 Hands-On Training

For more information about our company and SedNet products, visit our Web site at:

http://www.sederta.com

1. INTRODUCTION

1. 1 VME-400 Communication Adapter Features

1394

- 100, 200 400 Mbps data communication transfer
- Physical isolated interface
- Asynchronous and isochronous communication
- Three IEEE-1394 connector ports
- 1394 hot plug-and-play
- Up to 63 nodes may be connected along the same bus
- Fully compatible with SedNet™ adapters and libraries
- Limited compatibility with the IEEE-1394 (1995) standard:
  - Transaction layer for read and write operations
  - Basic space allocated for control and status registers (CSR)
  - Minimum configuration ROM

User Memory

- 256KB 32-bit Fast SRAM, 0-wait state
- 256KB 8-bit FLASH memory
- 32MB DRAM

Firmware

- Complete resident firmware, an intelligent bridge between 1394 and local buses.
- Real-time specific functions:
  - Analog and digital provider
  - Programmable isochronous timer to synchronize all attached nodes and IP modules
  - Specific real-time functions
- Capabilities for adding specific functions

IP Carrier Section

- Up to 12 Industrial Pack sites (32MHz or 8MHz) using 6U IP Module Carrier Boards (SD-VME-
- IPC or SD-VME-IPA) may be added
- Interrupt and DMA transfer supported
- Real-time synchronization between all attached devices and IP modules

Hardware Specifications

- Length: Standard VME 6U
- Storage temperature: 0°C to 70°C
- Operating temperature: 5°C to 55°C
- Operating humidity: 20% to 90% non-condensing
- Operating voltage: +5,+/-12V, +/-5%

1. 2 Basic Functionality Overview

1.2.1 Block diagram of the 1394-VME master adapter

The SedNet VME-400 master adapter is a high-performance intelligent bridge between the IEEE-1394 400 Mbps serial bus and Industrial Pack (IP) Modules. It contains a large amount of user-resident memory (FLASH, SRAM and DRAM), and has been implemented with an RS-232 port (serial port) for debugging and configuration purposes. The relationship between each of these components is illustrated below:

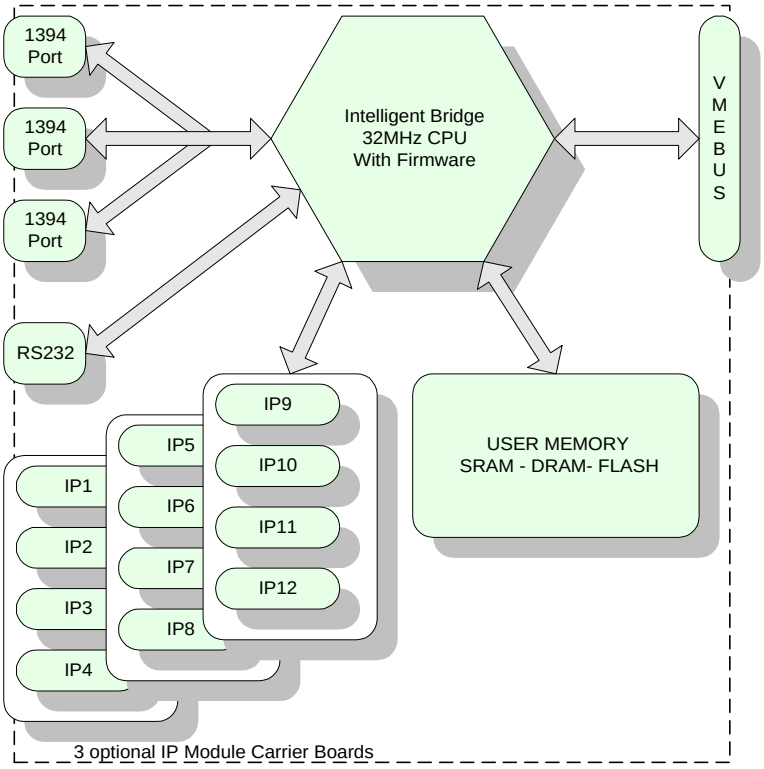


Figure 1 Block diagram of the 1394-VME master adapter

The VME-400 includes a 32-bit integrated processor unit, the MC68360, with an operating speed of 32 MHz. The processor combines high-performance data manipulation capabilities with powerful on-chip peripheral subsystems. The communication adapter includes 256KB of user EPROM FLASH memory, 256KB of user SRAM (static RAM), and 32MB of SIMM DRAM. The card interfaces up to twelve IP modules with the IP carrier board (SD-VME-IPC or SD-VME-IPA.)

The RS-232 port is mainly used for system configuration and debugging. It can be used to monitor system activity and is an easy way to reconfigure your system. You can also configure the VME-400 board via the 1394 bus.

1.2.2 Accessing the 1394-VME master adapter from the 1394 bus

The VME-400 board maps all its local memory space, including the user memory and IP module(s), on the 1394 bus.

The VME-400 board can be accessed by other 1394 devices to perform read or write operations to the local memory space.

Sending a 1394 request to the VME-400 performs these accesses. The main difference between the two accessed zones is the address offset high (16 bits) in the header of the 1394 packet, as shown in Table 1.

Address High	Destination Address
0x0000	Local memory space
0xFFFF	Space reserved for CSRs

Table 1 Destination address depending on the packet's address offset high

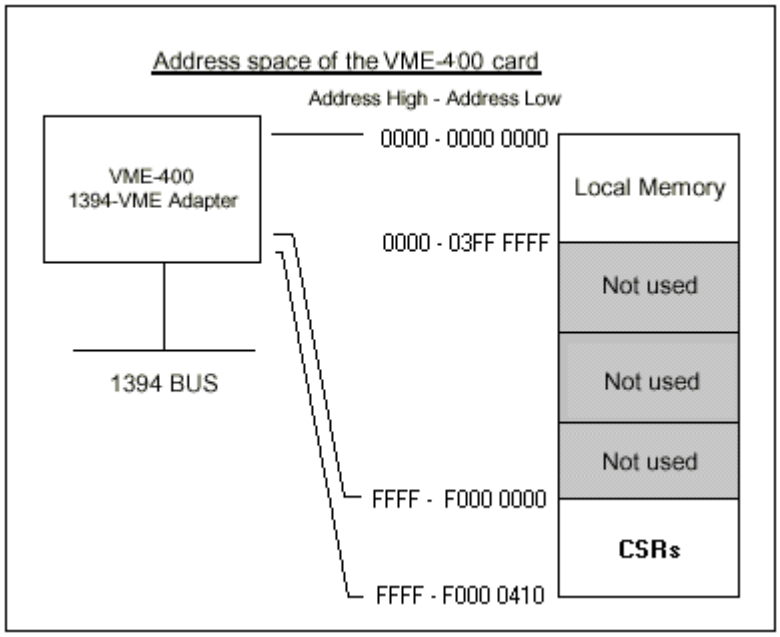


Figure 2 Memory of the VME-1394 bridge, as seen from the 1394 bus

1. 3 Architecture Overview

The VME-400 block diagram is shown in the figure below.

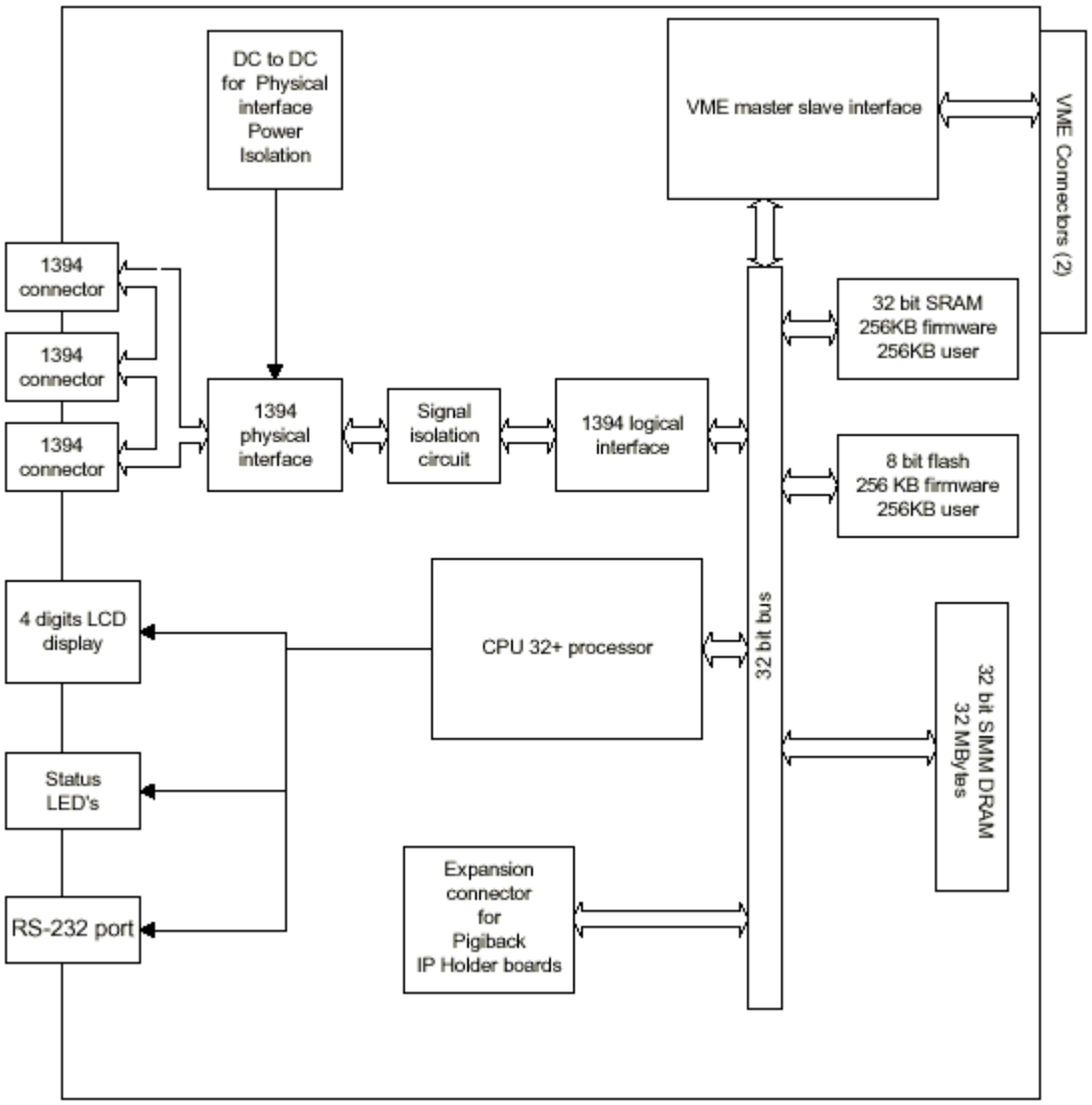


Figure 3 VME Block Diagram

1.3.1 MC68EN360 CPU

The MC68360 controller is part of the Motorola CPU32+ family. The CPU 32 main processor has a 32-bit data path and can address up to 4GB of memory space. Also present is an on-chip RISC processor that acts as a dedicated communications coprocessor for items such as DMA, serial communications and timers.

1.3.2 1394 Serial Bus Interface

The VME-400 uses the TSB12LV01 chip as a link layer to transmit and receive 1394 packets. It also generates and checks the 32-bit CRC. The link layer only provides a half-duplex (transmit or receive) data packet delivery service. The process of delivering a single packet is called a *subaction*. Two types of *subactions* are used in the serial bus link layer.

a) *Asynchronous subactions* – a variable amount of data and several bytes of transaction layer information are transferred to an explicit address and an acknowledge is returned
b) *Isochronous subactions* – a variable amount of data is transferred on regular intervals over reserved bandwidth. No acknowledge is returned.

The 1394 link layer has a software-adjustable FIFO for packet size and performance optimization. It interfaces directly with its companion chip, the TSB21LV03A physical layer. The TSB21LV03A can support bus speeds of 100, 200 and 400 Mbps.

The physical layer provides the analog transceiver functions needed to implement a three-port node in a cable-based 1394 bus. The three IEEE-1394 compliant cable ports provide a bandwidth of 400 Mbps. Its logic performs system initialization and arbitration functions.

1.3.3 Memory Blocks

The VME-400 communication adapter is implemented with the following memory units:

- 256 KB 8-bit program FLASH
- 256 KB 8-bit user FLASH
- 256 KB 32-bit program SRAM
- 256 KB 32-bit user SRAM
- 32 MB 32-bit (SIMM) user DRAM

1.3.4 4-Digit LCD Display

The LCD display is used to indicate the system status. Messages are displayed during hardware testing or when a system fault occurs. The LCD display is also used to display the adapter board's 1394 node number.

1.3.5 Status LED

The card's front panel has been outfitted with four LEDs:

- Run
- Idle
- Error
- SCon

The **Run LED** is lit when the 1394 interface is active. It indicates that a 1394 packet is currently being sent or received.

The **Idle LED** is lit when neither the 1394 nor the VME bus is currently being accessed.

The **Error LED** flashes when an unrecoverable error has occurred. The processor will then be in a HALT state. When a minor error has occurred, the LED will be ON until the cause of the error has been deactivated. If the communication adapter is being used under normal conditions, the Error LED should be OFF at system RESET.

1.3.6 RS-232 Port

Using the proper serial port interface, the RS-232 port is used as a debugging tool for system functionalities such as 1394 quadlet write and read requests/responses. It can also be used to configure system parameters such as the 1394 node number and the card's VME mapping.

For more information about the VME-400 RS-232 port, refer to the EVM-400-SI manual.

2. SYSTEM CONFIGURATION

2. 1 SedNet Adapter Card Boot-Up Sequence

At system start up, the name ***SedNet*** will appear on the LCD display, followed by the software version number.

A sequence of five tests is then executed:

- Test 1 verifies the user SRAM. If an error is detected while executing the test, "ERR1" will be written on the LCD display located on the front panel of the adapter.

- Test 2 verifies the DRAM. The LCD display will show the amount of DRAM memory available in the system: either 0 MB or 32 MB. The message "ERR2" will be displayed on the LCD if this test fails.

- Test 3 verifies the 1394 adapter. It will test and initialize the 1394 protocol's physical and link layers. An "ERR3" message will be displayed on the LCD if an error is detected while executing this test.

- Test 4 verifies the VME interface. An "ERR4" message will be displayed on the LCD if an error is detected while executing this test.

Note: Refer to ANNEX V: Error Messages for a description of the error messages.

After these tests have been completed successfully, the LCD display will show the 1394 bus and node number of the VME-400 card. When one or several 1394 connectors are connected to the communication adapter, a vertical line will appear beside one of the last three digits of the LCD display. One line should appear for each connector that was successfully detected, showing the position of the detected connector.

Example: If the 1394 bus number is `0x3ff` and the node number is `0x001`, the LCD will display `FFC1` in hexadecimal. If there is a 1394 connector in the 1394 sockets of the card, a vertical bar will be displayed near the second digit of the LCD display.

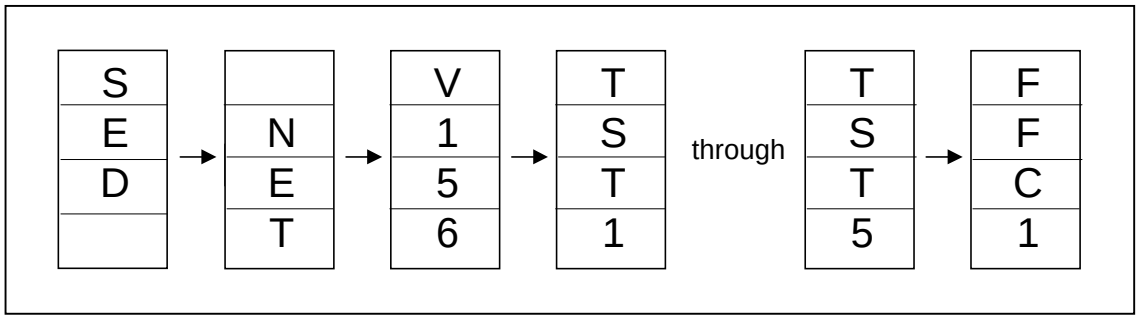


Figure 4 Example of a boot-up sequence on the LCD display

2. 2 SedNet Adapter Card Jumper Configuration

Two jumpers have been added to the VME communication adapter: one for the power supply capabilities of the 1394 connectors when the VME is not powered on (P8) and the other to indicate the VME bus controller capability (P10). These jumpers are located on the adapter card as indicated in the diagram below:

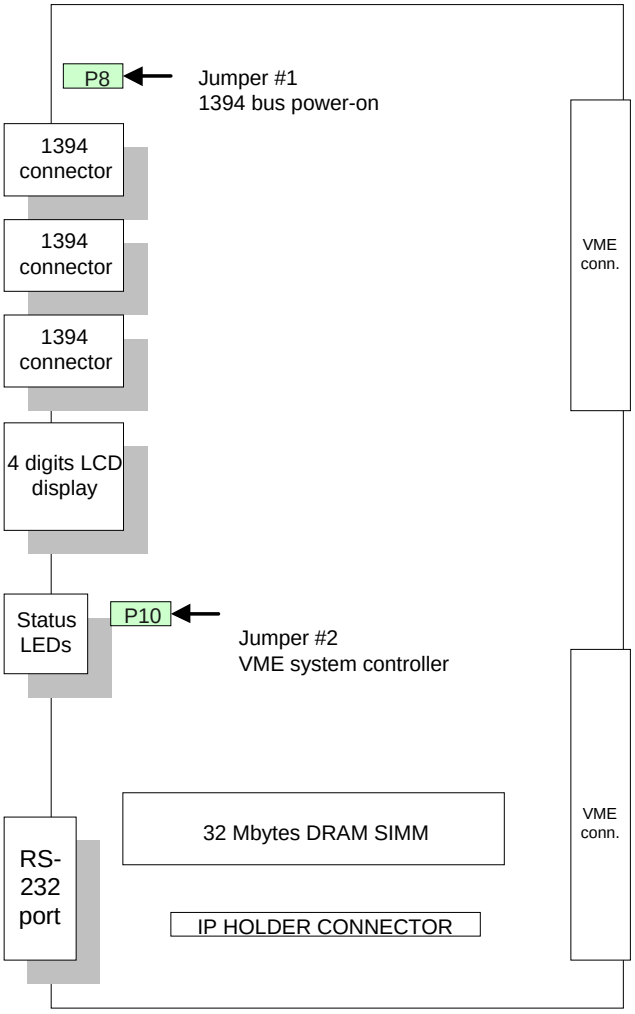


Figure 5 Position of the jumpers on the card

P8 is used to indicate whether the PHY (1394 physical layer) can be powered by the 1394 bus. When the jumper is connected, the PHY can receive power from the 1394 bus even if the VME bus is off.

When jumper P10 is connected, the adapter board can act as the VME system controller and perform VME bus arbitration.

2. 3 SedNet Adapter Card Parameters Configuration

Certain parameters must be set for a fully-functional system. All parameters should normally be set up at system delivery. If, for any reason, you wish to change any of these settings, follow the instructions below. (For a quick reference of the parameters and how to modify them, see ANNEX I: SRAM parameters and ANNEX II: IMPLEMENTED CSRS at the end of this document.)

2.3.1 Parameter List

The key parameters are:

- 1394 Vendor ID
- IEEE-1394 Compliance
- Serial Number
- Version Number
- Bus Node and Bus Number
- Response Flag
- RS-232 port configuration flag

The other parameters are outlined in ANNEX I: SRAM parameters at the end of this document.

1394 Vendor ID:

This parameter is used to indicate the identification number of the adapter card's manufacturer. Each manufacturer has a distinct serial number. The vendor ID is usually written once and never modified afterwards.

IEEE-1394 Compliance:

This parameter indicates whether the communication adapter is being used in SedNet-compatible mode or IEEE-1394 compliant mode.

Serial Number:

This parameter is used to indicate the serial number of the adapter card. Each adapter card has a distinct serial number. This is how you can tell your communication adapters apart. The serial number parameter is usually written once and never modified afterwards.

Version Number:

This parameter is used to give the software and hardware version of the adapter board. It is mostly used to identify the current software version on the card when a software update is released. This is how to read the version information:

31	Hardware Version	16	15	Software Version	0

Bus Node and Bus Number:

The node number is the number assigned to the 1394 device to uniquely identify it for the purposes of the 1394 bus configuration. The bus number is used in cases of multiple 1394 buses, for example, there could be two nodes assigned #2, one on bus 1 and another on bus 2.

Both the bus node and the bus number are set to 0 by default. The bus number can take values between 0 and 1023; whereas the node number can take values between 0 and 63. This parameter does not apply if the communication adapter's IEEE-1394 compliance parameter is set to 1 (IEEE-1394 compliance = TRUE) because in this case the node number used for the bus is the device's physical ID, as assigned by the root master upon bus initialization.

31	Bus Number	22	21	Node Number	16	15	Reserved	0

Response Flag:

This parameter tells the system whether or not it must deliver an acknowledgement for every read or write request being sent. When the acknowledge is withheld, more packets can be sent in a shorter time span; however, there is no way to verify if the packets have been properly sent.

These are the possible value for the parameter:

Value	Status
0	No Response
1	Send a Response

Table 2 Configuration values for the Response flag

RS-232 Configuration Flag:

This parameter tells the system whether the RS-232 port (serial port) is being used as a regular RS-232 port for data transfer or as a system debugger. If it is being used for debugging, the port will be connected to a terminal to monitor system activity, or configure and test the system. The default value is 0, which means that the port is being used for debugging purposes.

There are two possible values for this parameter:

Value	Mode
0	Debugging mode
1	Standard RS-232 port

Table 3 Configuration values for the RS-232 port

2.3.2 Modifying Parameters

It is possible to change a parameter by writing to its assigned memory space. Unfortunately, this type of update is only temporary. On the other hand, if the new parameter values are written to the FLASH memory, they will be permanently retained (or at least until they are changed again) and will be updated automatically after the operation. Follow the steps below to modify parameters in the FLASH memory:

Action	Address	Data
1. Write the new parameter value to SRAM.	The addresses of the parameter.	The new parameter values.
2. Write the update parameter semaphore to begin copying to the FLASH memory.	`0x000AFF88`	`0x00000001`

Table 4 How to update the system parameters

2.3.3 Parameter Addresses

Refer to ANNEX I: SRAM parameters for details.

Parameter	Address
IEEE_1394_VENDOR_ID	0x000AFEDC
CHIP_ID_HIGH	0x000AFEE0
CHIP_ID_LOW	0x000AFEE4
IEEE_1394_COMPLIANCE	0x000AFEE8
MAX_REC	0x000AFEEC
ISOCH_ENABLE	0x000AFEF0
BUS_MANAGER_ENABLE	0x000AFEF4
ISOCH_MANAGER_ENABLE	0x000AFEF8
CYCLE_MASTER_ENABLE	0x000AFEFC
SERIAL_NUMBER	0x000AFF00
VERSION_NUMBER	0x000AFF04

BUS_NODE	0x000AFF08
RESPONSE_FLAG	0x000AFF0C
ISO1_CHANNEL_NUM	0x000AFF10
ISO2_CHANNEL_NUM	0x000AFF14
CS5_ASIZ (used for VME configuration)	0x000AFF18
SLAV_ADD (used for VME mapping)	0x000AFF1C
SLAV_ADD_MSK (used for VME mapping)	0x000AFF20
VME_ADD (used for VME mapping)	0x000AFF24
VME_ADD_MSK (used for VME mapping)	0x000AFF28
1394 Packet length and packet address high (only used when accessing 1394)	0x000AFFAC
1394 Packet address low (only used when accessing 1394)	0x000AFFB0
1394 Response packet address high (only used when accessing 1394)	0x000AFFB4
1394 Response packet address low (only used when accessing 1394)	0x000AFFB8
RS-232 Port Configuration	0x000AFFBC

Table 5 Addresses for the system parameters

2. 4 Activating the Data Transfer

Here are the steps you need to follow to send a 1394 packet from the VME-400 adapter:

NOTE: you must take control of the 1394 interface before you write the 1394 request. The 1394 request can be written at the address you want in user SRAM or in DRAM.

Action	Address	Value
1. Wait until the data at address 0x000AFFA8 is 0x0000, meaning that the 1394 interface is idle.	0x000AFFA8	0x00000000
2. Write 0x01 to the 1394 semaphore to take control of it.	0x000AFFA8	0x00000001
3. Write the 1394 request packet.	The address you select	Your 1394 Request Packet
4. Write the address of the 1394 packet you wrote in Step 3.	0x000AFFB0	Address of the 1394 request packet from Step 3
5. Write the length of the packet in the 16 most significant bits at address 0x000AFFAC.	0x000AFFAC	|32 length 16|15 origin 0|

6. Write the origin of the packet in the 16 least significant bits of address 0x000AFFAC. (Write a 0 if on the card.)	0x000AFFAC	|32 length 16|15 origin 0|
7. Write the address destination offset low of the response packet.	0x000AFFB8	Address of the 1394 response packet to be received
8. Write the address destination offset high of the response packet. (Write 0x0 if on the card.)	0x000AFFB4	Address offset high of the response packet to be received
9. Start the transfer by writing 0x02 at address 0x000AFFA8.	0x000AFFA8	0x00000002
10. Wait until the data at address 0x000AFFA8 is 0x0, meaning that the transfer is over.	0x000AFFA8	0x000000000
11. Retrieve the 1394 response packet at the address specified in Step 6-7.	See steps 6-7 above	1394 Response Packet

Table 6 Steps to activating 1394 packet transmission

Example:

Pick up a packet located at address 0x000A0000 with a packet length of 64 bytes. Then a write response will be expected at address 0x000C0000 on the board once the request has been made.

1- Wait until the 1394-VME master adapter becomes available (when the data at 0x000AFFA8 is equal to 0x0).

2- Request the 1394 bridge by writing 0x1 at address 0x000AFFA8.

3- Write the 1394 request packet at address 0x000A0000 on the VME-400 board.

4- Write 0x000A0000 at address 0x000AFFB0 to indicate where the packet must be retrieved.

5- Write 0x00400000 at address 0x000AFFAC.

→ 0040 is the length and 0000 means that the request is directly on the card.

6- Write 0x00000000 at address 0x000AFFB4.

7- Write 0x000C0000 at address 0x000AFFB8 (This is the acknowledge address.)

8- Write 0x00000002 at address 0x000AFFA8 to initiate the packet transfer.

9- Wait for the response packet at address 0x000C0000 on the board until the value at 0x000AFFA8 is equal to 0x0.

3. THE RS-232 SERIAL PORT

3. 1 Overview of the RS-232 Port

The RS-232 port on the VME-400 communication adapter serves many purposes:

- It can be used as a debugging tool to read and write memory blocks, configure the system parameters, update the card's firmware or perform software reset of the system.

- It can also be used as an RS-232 port to connect to a printer, modem or any serial communication device. (Not currently available.)

- It can be connected to a terminal in order to monitor 1394 packets on the card and give VME access to the card. Memory blocks can be read and some software functions (semaphores) can be used with this option. (Not currently available.)

Refer to the Serial Interface manual for more information about the RS-232 port and how to use it.

3.1.1 Debug Mode

In debug mode, no software is running on the card. Memory reads and writes can be performed, as well as memory block fills. The system configuration can be accessed or modified from this option. New versions of the firmware can be uploaded to the FLASH (greatly simplifying software upgrades) and 1394 accesses can be made for testing purposes.

3.1.2 RS-232 Port Mode

In RS-232 mode, the port is used as an input/output port for serial communication. No special software or hardware is required to communicate with the port, and any RS-232 device can be accessed via the serial port. The only settings that might require changing are the communication parameters such as baud rate, data bits, parity, start and stop bits.

3.1.3 Monitor Mode

In monitor mode, the port acts as a window where the user can view the packets being sent to and from the system. It can also be used to read the system's memory; however, no writes are permitted since this could put the system in an unstable state. Certain software functions (semaphores) can be used from this option and the system configuration parameters are always available to the user.

4. HOW TO UPDATE YOUR CARD'S FIRMWARE

4. 1 Updates from the RS-232 Port

Sederta's specially-designed firmware update application is a Windows 95/98/NT program included in your package when you purchase a SD-EVM-400 adapter card. The program is named `EVM-400-SI` and is used via the host computer's serial port. For more information about the RS-232 port and how to use it, refer to the *EMV Serial Interface* manual. It is available in printed format with this package, as well as online in the Help menu of the `EVM-400-SI` program.

Follow the steps below to update your VME card's firmware using the `EVM-400-SI` application:

1. Launch the `EVM-400-SI` application.
2. Choose *Update VME* from the *Update* pull-down menu. The *Update VME* dialogue box will appear.
3. Indicate the bus and node ID of the card whose firmware you wish to update in hexadecimal.
4. Select the *Motorola S-Record Format* file with the file extension *.DWN* containing the update to be written to the firmware of the selected VME communication adapter. You are provided with this file when an updated version of the firmware is released. When you click the *File name* textbox, a dialogue box will appear where you select the appropriate *DWN* file. The truncated path of the file you choose will be displayed in this textbox once the dialogue box closes.
5. Click the *Send* button to download the firmware update information to the VME communication adapter you selected in *Update VME* dialogue box.
6. Click the *Quit* button to return to the main screen of the `EVM-400-SI` program.

When performing the firmware update to the FLASH, the LCD display will show the percentage progress of the update once the DWN file has been fully loaded on to the card's SRAM. After the update has been completed, the LCD display should show the 1394 bus and node number again. If this is not the case, an error has occurred while updating the firmware.

Updating the firmware of your VME communication adapter will monopolize your system's resources until completion of the procedure. You must wait until the process is over before attempting to use the system for other purposes.

Note: After updating the firmware, you must RESET the card for the modifications to take effect. (See ANNEX III for the address of the semaphore that resets the communication adapter.)

Caution: Performing a FLASH update may cause loss of all data in the user SRAM because of the card reboot process.

5. ISOCHRONOUS TRANSACTIONS

The isochronous digital provider manages the transfer of data blocks at regular intervals (isochronous cycles on the 1394 bus.) The isochronous digital provider can be activated by an application to transmit frames.

The isochronous digital receiver manages the reception of data blocks on regular intervals. A SD-VME-400 communication adapter can be configured to receive data from a maximum of two channels. The receiver stores dedicated packets in memory. Refer to the IEEE-1394 standard for more information.

5. 1 ISOCHRONOUS DIGITAL PROVIDER

The isochronous digital provider sends isochronous packets at a regular time interval. The parameters used to configure the isochronous digital provider are defined by the following constants:

	Address Location	Name	Definition	Default Value
1	0xb0010	ISO_PROVIDER	0 = Idle 1 = Start 2 = Running 3 = Stop	0 = Idle
2	0xb0014	ISO_PROV_RATIO	Transfers a block every *n* cycles.	1 = Every cycle
3	0xb0018	ISO_PROV_HEADER	TAG: Format of the data carried (bits 14-15) chanNum: Number of the channel transmitting packets (bits 8-13) spd: Speed of the IEEE-1394 bus (bits 4-5): 00 = 100Mbps 01 = 200Mbps 10 = 400Mbps sy: Transaction layer-specific synchronization bits. 0 = no synchronization (bits 0-3)	TAG = 1 chanNum = 0 spd = 01 (200Mbps) sy = 1
4	0xb001c	ISO_PROV_BLOCK_LENGTH	Length of the block of memory to be written, expressed in bytes.	8 bytes (0 to 512 bytes)
5	0xb0020	ISO_PROV_FRAME_LENGTH	Length of the frame to be written, expressed in bytes.	512 bytes (0 to 2MB)
6	0xb0024	ISO_PROV_SRC_HIGH	Address high of the block of memory to be written.	0 (16 bits)
7	0xb0028	ISO_PROV_SRC_LOW	Address low of the block of memory to be written.	0x1000000 (32 bits)
8	0xb002c	ISO_PROV_PTR	Ptr of the address after a DMA transfer. Used to know where the pointer of the provider is.	0
9	0xb0030	ISO_PROV_DONE	Transmission of the block is completed when this bit is set to 1.	0
10	0xb0034	RESERVED		

11	0xb0038	ISO_PROV_CLK_SRC	Clock Source 0 = Off 1 = Cycle master (on-board) 2 = External clock	1 = Cycle master (on-board)
12	0xb003c	ISO_PROV_START_CYCLE	Forces the board to transmit cycle-start packets for cycle master node 0 = Idle 1 = Start 2 = Running 3 = Stop	0 = Idle

Table 7 Isochronous digital provider parameters

5.1.1 Configuring the isochronous provider

Configure the isochronous provider using the steps listed below:

- Set the ratio in the ISO_PROV_RATIO constant.
- Configure the header in the ISO_PROV_HEADER constant by setting values for TAG, chanNum, spd and sy

  TAG: Isochronous data format tag. (See *IEEE 1394-1995 6.2.3* and *IEC 61883* for more information about isochronous data format tags.)

  sy: Identifies the synchronization.
- Set the length of the blocks to be transferred in the ISO_PROV_BLOCK_LENGTH constant.
- Set the length of the frames to be transferred in the ISO_PROV_FRAME_LENGTH constant.
- Configure the data source in the ISO_PROV_SRC_LOW and ISO_PROV_SRC_ HIGH constants.

Enable the provider by setting the ISO_PROVIDER constant to 1.

If a cycle master is present on the bus, transmission of isochronous packets will begin.

If no communication adapter has been configured as the cycle master, you can use a semaphore to start the cycle master. It is crucial to activate the ISO_PROV_START_CYCLE semaphore on the ROOT adapter so that the cycle master is enabled. To start the cycle master, set ISO_PROV_START_CYCLE to 1.

Before updating the data to be transmitted, you should check if the data has already been transferred. The ISO_PROV_PTR constant will show the starting point of the next block to be transmitted. This value is updated following the transfer of each block. Similarly, ISO_PROV_DONE is also set to 1 following each block transfer.

An example of how to configure and implement the isochronous provider is given on the next page.

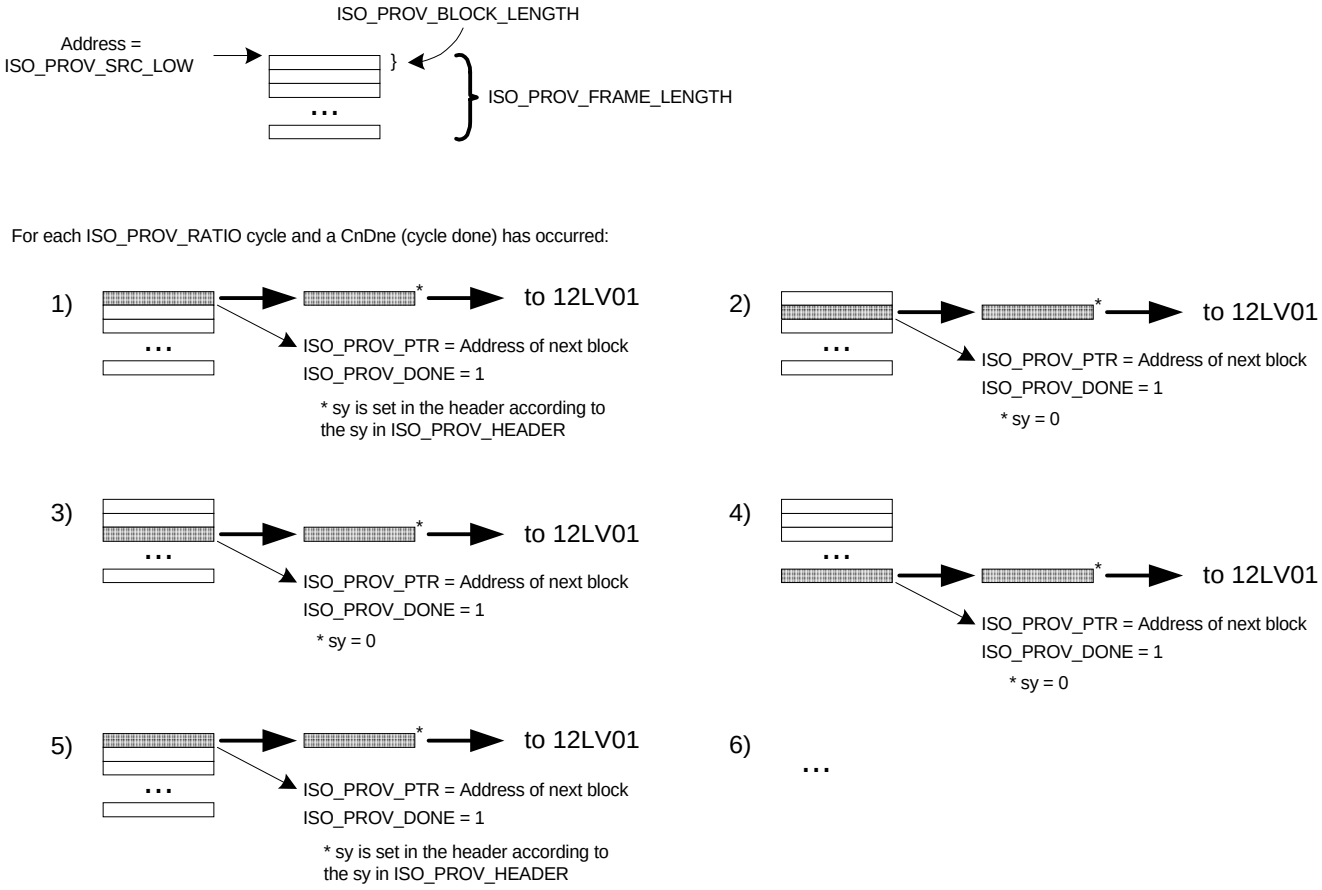


Figure 6 Configuring and implementing the isochronous provider

5. 2 ISOCHRONOUS DIGITAL RECEIVER

The isochronous digital receiver receives automatic isochronous block write request packets from the isochronous digital provider or any device that can send 1394 packets. The parameters used to configure the isochronous digital receiver are defined by the following constants:

	Address Location	Name	Definition	Default Value
1	0xaff34	ISO_REC_ENABLE_OFF	Enables the receiver channel 0 = All channels disabled 1 = Channel 1 enabled 2 = Channel 2 enabled 3 = Both channels are enabled	0 = All channels disabled
2	0xaff10	ISO_REC_CHAN_1_OFF	TAG: Format of data carried (bits 6-7) chanNum: Isochronous receive port 1channel number (bit 0 to 5)	TAG = 1 chanNum = 0

3	0xaff38	ISO_REC_BLOCK_LENGTH_1_OFF	Length of the block of the memory to receive in byte (channel 1)	8 bytes (0 to 512 bytes)
4	0xaff3c	ISO_REC_FRAME_LENGTH_1_OFF	Length of the frame of the memory to be received, expressed in bytes (channel 1).	512 bytes (0 to 2Mb)
5	0xaff40	ISO_REC_DEST_HIGH_1_OFF	Address high of the block of the memory to be written (channel 1).	0 (16 bits)
6	0xaff44	ISO_REC_DEST_LOW_1_OFF	Address low of the block of the memory to be written (channel 1).	0x1200000 (32 bits)
7	0xaff48	ISO_REC_PTR_1_OFF	Ptr of the address after a DMA transfer. Used to know where the pointer of the receiver is (channel 1).	0
8	0xaff4c	ISO_REC_SYNC_1_OFF	syMode: The type of synchronization (bits 4-7) 0 = no synchronization 1 = synchronization for each corresponding sy sy: transaction layer-specific synchronization bits. 0 = no synchronization (bits 0-3)	syMode = 0 sy = 1
9	0xaff50	ISO_REC_DONE_1_OFF	Reception of the block has been completed when this bit is set to 1.	0
10	0xaff14	ISO_REC_CHAN_2_OFF	TAG: Format of data carried (bits 6-7) chanNum: Isochronous receive port 2 channel number (bits 0-5)	TAG = 1 chanNum = 1
11	0xaff54	ISO_REC_BLOCK_LENGTH_2_OFF	Length of the block of memory to be received, expressed in bytes (channel 2).	8 bytes (0 to 512 bytes)
12	0xaff58	ISO_REC_FRAME_LENGTH_2_OFF	Length of the frame of memory to be received, expressed in bytes (channel 2).	512 bytes (0 to 2MB)
13	0xaff5c	ISO_REC_DEST_HIGH_2_OFF	Address high of the block of the memory to be written (channel 2).	0 (16 bits)
14	0xaff60	ISO_REC_DEST_LOW_2_OFF	Address low of the block of the memory to be written (channel 2).	0x1400000 (32 bits)
15	0xaff64	ISO_REC_PTR_2_OFF	PTR of the address after a DMA transfer. Used to know where the pointer of the receiver is (channel 2)	0
16	0xaff68	ISO_REC_SYNC_2_OFF	syMode: The type of synchronization (bits 4-7) 0 = no synchronization 1 = synchronization for each corresponding sy sy: transaction layer-specific synchronization bits. 0 = no synchronization (bits 0-3)	syMode = 0 sy = 1
17	0xaff6c	ISO_REC_DONE_2_OFF	Reception of block has been completed when this bit is set to 1.	0

Table 8 Isochronous digital receiver parameters

5.2.1 Configuring the isochronous receiver

Configure the isochronous receiver using the steps listed below:

- Configure the reception channel in the `ISO_REC_CHAN_x_OFF` constant:
  `TAG`: Isochronous data format tag. (See *IEEE 1394-1995 6.2.3* and *IEC 61883* for more information about isochronous data format tags.)
  Only packets with the same `TAG` will be accepted on this channel.
- Set the length of the blocks to be received in the `ISO_PROV_BLOCK_LENGTH_x_OFF` constant.
- Set the length of the frames to be received in the `ISO_PROV_FRAME_LENGTH_x_OFF` constant.
- Set the destination of the data blocks in the `ISO_PROV_SRC_LOW_x_OFF` and `ISO_PROV_SRC_HIGH_x_OFF` constants.
- Configure the synchronization in the `ISO_REC_SYNC_x_OFF` constant:
  syMode: the synchronization type: 0 = no synchronization
  1 = synchronization for each corresponding sy
  sy: transaction layer-specific synchronization: 0 = no synchronization

Activate the receiving channel(s) in the `ISO_REC_ENABLE_OFF` constant.

All these parameters can be saved in the user FLASH.
NOTE: For the card to be standalone, they must be saved here.

Before reading the data, it is important to check if it has been received. The `ISO_REC_PTR_x_OFF` constant indicates where in memory the next bock will be received. This value is updated every time a block is received. The `ISO_REC_DONE_x_OFF` constant is also set to 1 after each block is received.

Examples of configuring and implementing an isochronous digital receiver appear on the following page.

The following is an example of the isochronous digital receiver with the syMode in `ISO_REC_SYNC_x_OFF` equal to 1.

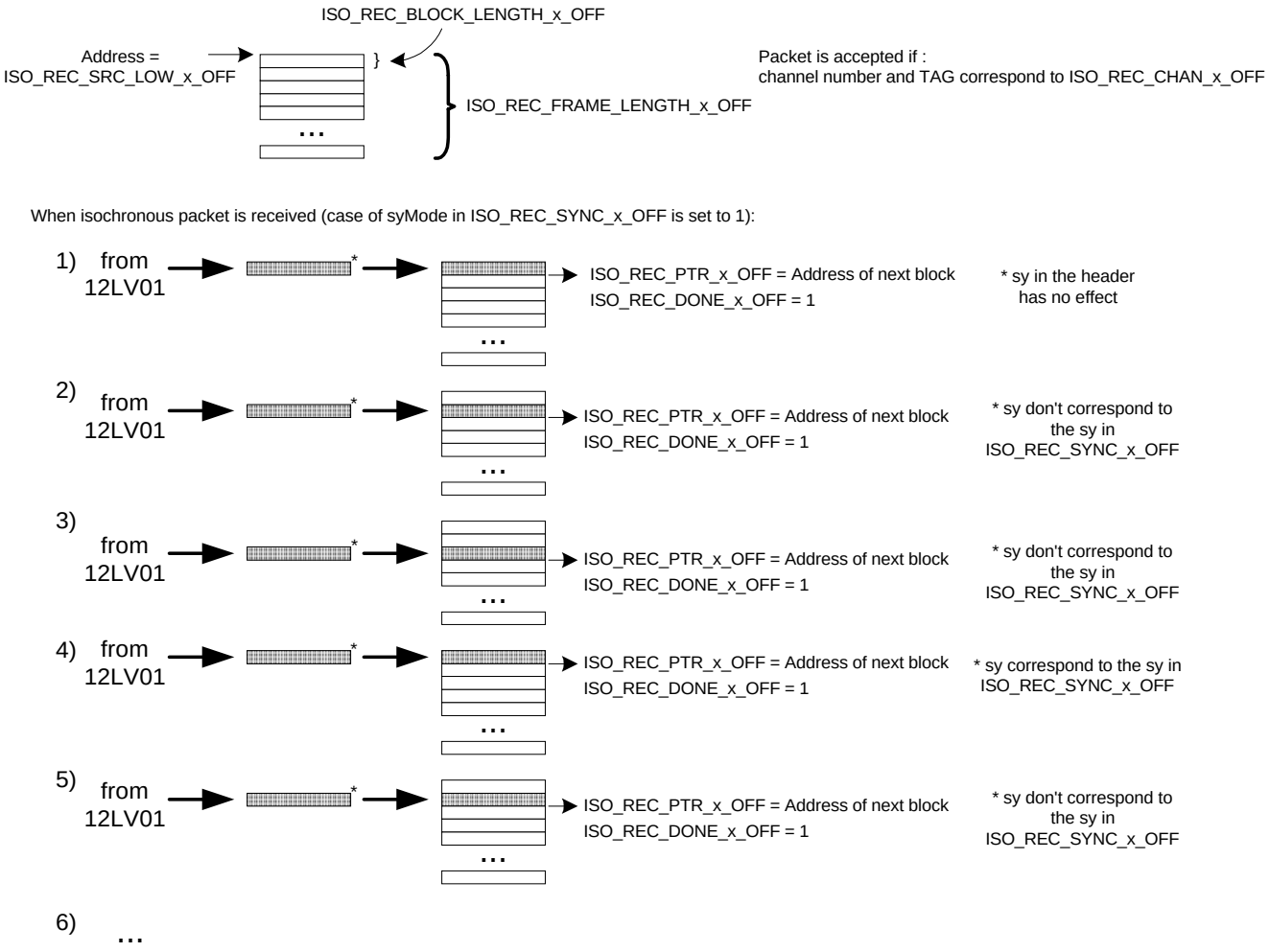


Figure 7 Configuring and implementing the isochronous digital receiver with the syMode = 1

The following is an example of the isochronous digital receiver with the syMode in `ISO_REC_SYNC_x_OFF` equal to 0.

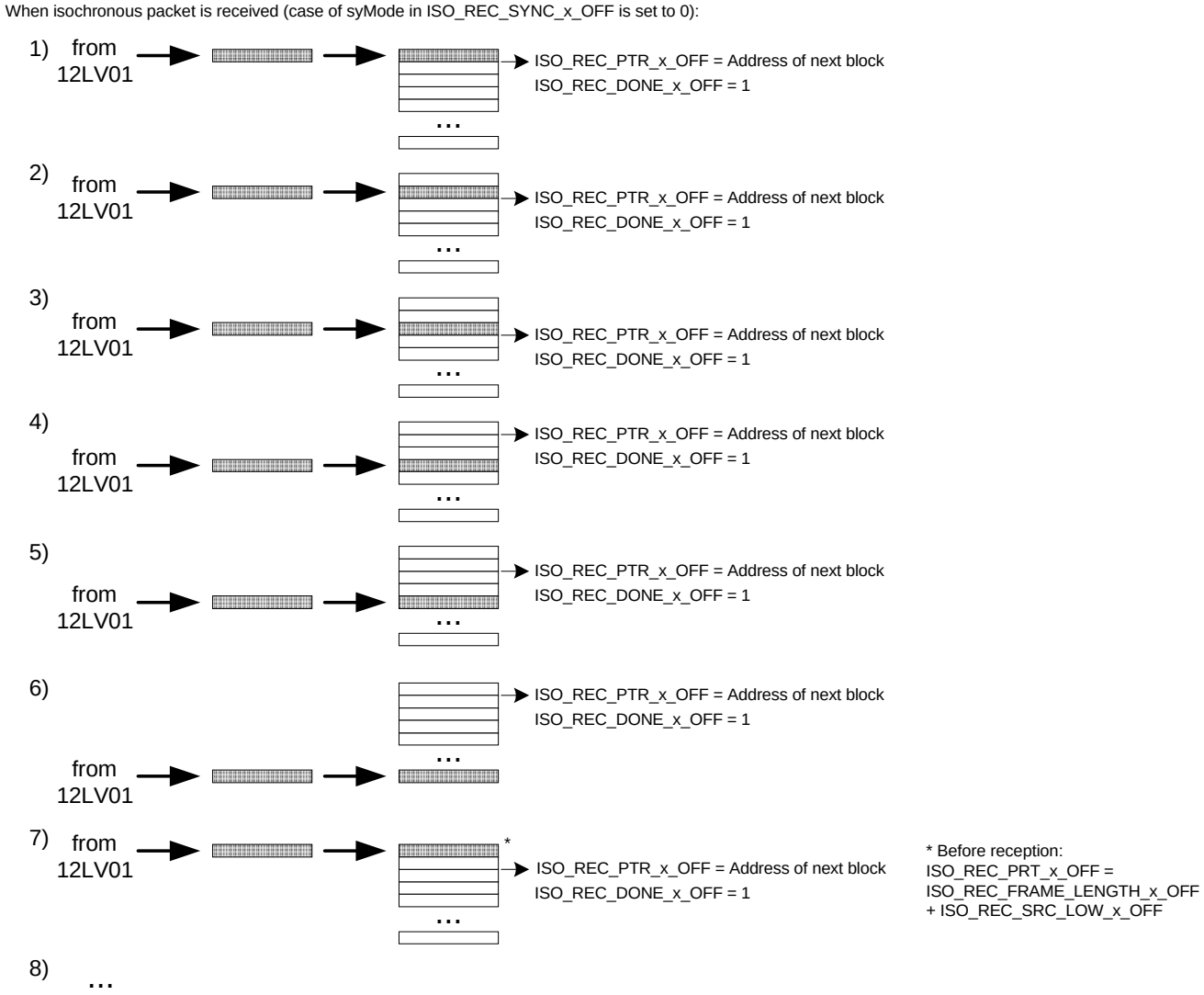


Figure 8 Configuring and implementing the isochronous digital receiver with syMode = 0

6. IP CARRIER BOARD DESCRIPTION

6. 1 Introduction

Two IP carrier boards are currently available:

- SD-VME-IPC
- SD-VME-IPA

These IP carrier boards are stacked on the VME-400 board. Up to three IP carrier boards may be stacked on the VME-400 board at any one time.

6.1.1 SD-VME-IPC

This IP carrier board can support up to 4 IP modules for I/O transactions on the 16-bit IP Bus. The IP carrier board supports operating speeds of either 8 or 32 MHz. Setting a jumper selects the speed to use.

Four 50-pin DIN connectors are placed on the front panel of the board. Each connector corresponds to one of the four IP Modules on the carrier board.

6.1.2 SD-VME-IPA

The SD-VME-IPA carrier board has the same functionality as the SD-VME-IPC board. The main difference is the type of the connectors placed on the front panel of the board.

Three DB37 connectors are placed on the front panel. The first connector is shared between IP Modules 2 and 3. The two remaining connectors are joined together and shared between IP Modules 0 and 1.

6. 2 Adapter Block Diagrams

6.2.1 SD-VME-IPC Block Diagram

The SD-VME-IPC block diagram is shown below.

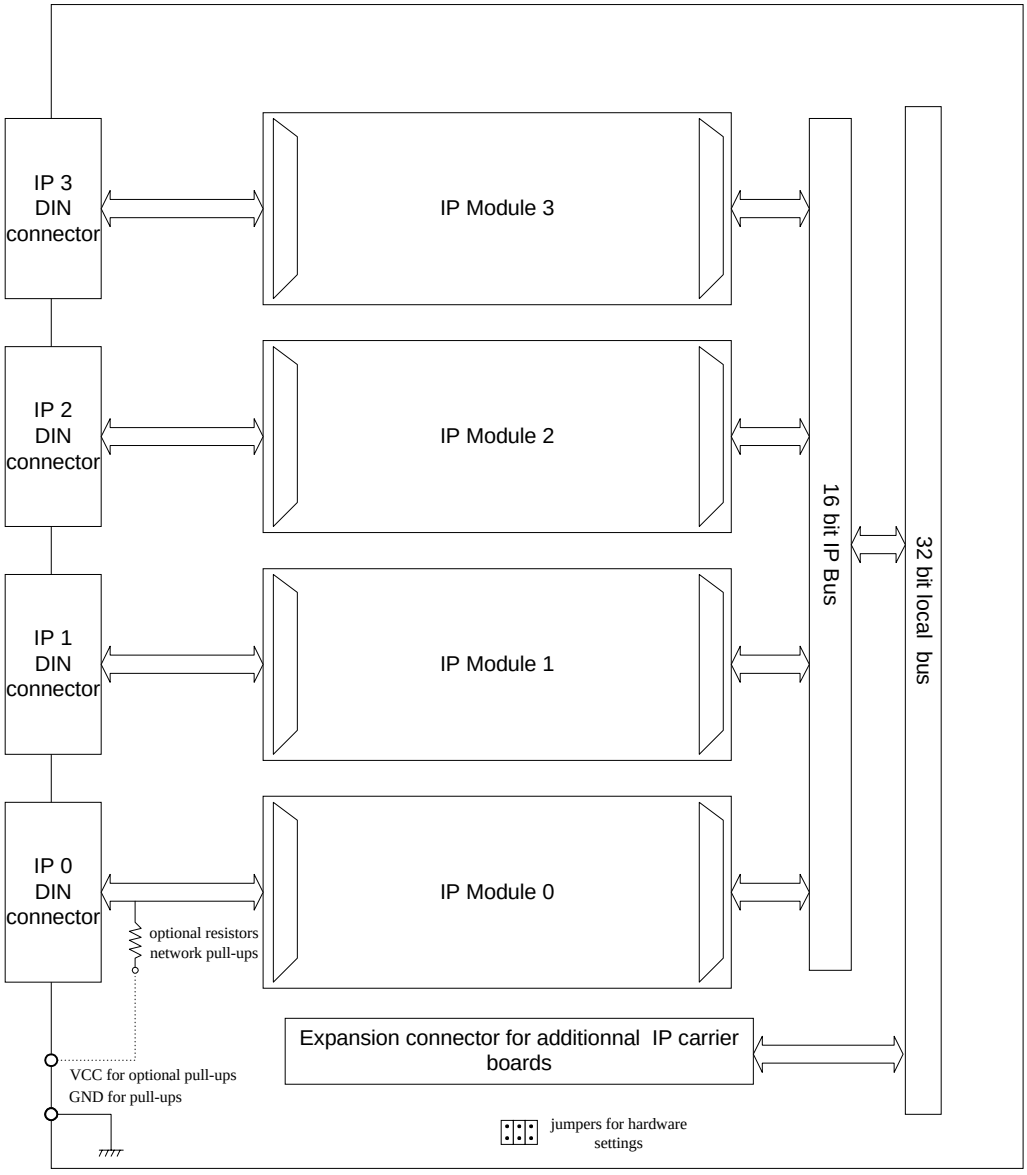


Figure 9 Block diagram of the SD-VME-IPC carrier board

The four DIN connectors are 50-pin connectors with a pin-to-pin correspondence with the IP Modules' pins. The pin correspondence of the connector is explained in Table 9.

DIN connector pin	IP Module pin
1	I/O 1
2	I/O 2
3	I/O 3
..	..
49	I/O 49
50	I/O 50

Table 9 Pin-out correspondence between DIN connector and IP Modules

The on-board jumpers are used to set hardware options. One jumper is used to set the board's clock rate (8MHz or 32MHz), and the others are used to configure the carrier board ID number, as illustrated below.

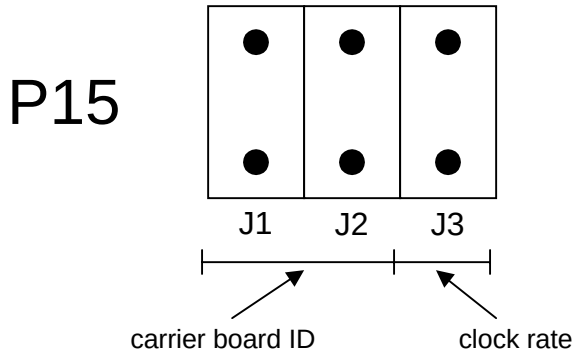


Figure 10 Jumper functions

The jumper settings are explained in the following two tables, where an 'X' indicates that the jumper is installed and a dash indicates that no jumper is installed.

P8	P10	Board ID
X	X	IP carrier 1
-	X	IP carrier 2
X	-	IP carrier 3
-	-	Must not be used

Table 10 Jumper settings for the board ID

P10	Clock Rate
	8 MHz
X	32 MHz

Table 11 Jumper settings for Clock Rate.

Optional resistor networks can be inserted on the board to provide pull-ups for the IP Module's I/O. When the pull-ups are used, they must have VCC provided for them, depending on IP Module types and requirements.

6.2.2 SD-VME-IPA Block Diagram

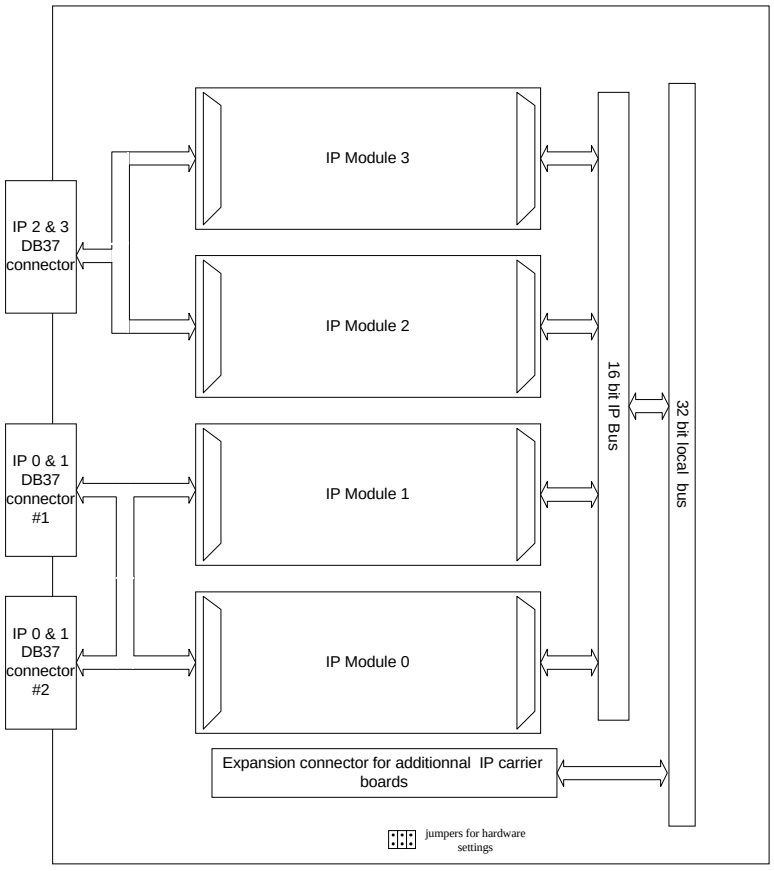


Figure 11 Block diagram of the SD-VME-IPA carrier board

The three DB37 connectors are shared between the various IP Modules. One connector is used for IP Modules 2 and 3, and the other two are redundant and used for IP Modules 0 and 1. Table 12 and Table 13show the pin correspondence of the connector.

DB37-1 pin	IP Module-2 pin	IP Module-3 pin
1	-	I/O 1
Through	-	Through
16	-	I/O 16
17	I/O 1	-
Through	Through	-
32	I/O 16	-
33-37	GND	

Table 12 Pin-out correspondence between DB37 connector and IP Modules 3 and 4

DB37-2 and DB37-3 pin	IP Module-0 pin	IP Module-1 pin
1	-	I/O 1
Through	-	through
16	-	I/O 16
17	I/O 1	-
Through	through	-
32	I/O 16	-
33-37	GND	

Table 13 Pin-out correspondence between DB37 connector and IP Modules 1 and 2

The on-board jumpers are used to set hardware options. One jumper is used to set the adapter's clock rate (8MHz or 32MHz), and the others are used to configure the carrier board ID number, as explained in the figure below.

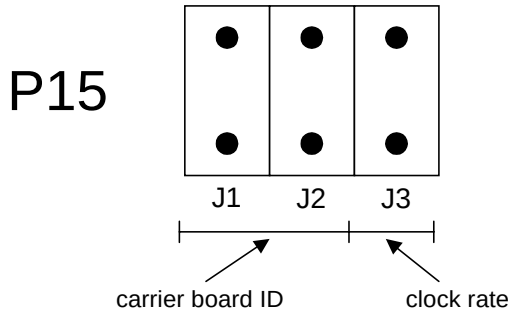


Figure 12 Jumper functions

The jumper settings are outlined in the two tables below, where an 'X' indicates that the jumper is installed and a dash indicates that no jumper is installed.

P8	P10	Board ID
X	X	IP carrier 1
-	X	IP carrier 2
X	-	IP carrier 3
-	-	Must not be used

Table 14 Jumper settings for the adapter board ID

P10	Clock Rate
	8 MHz
X	32 MHz

Table 15 Jumper settings for Clock Rate

6. 3 Front Panel View

6.3.1 SD-VME-IPC Front Panel View

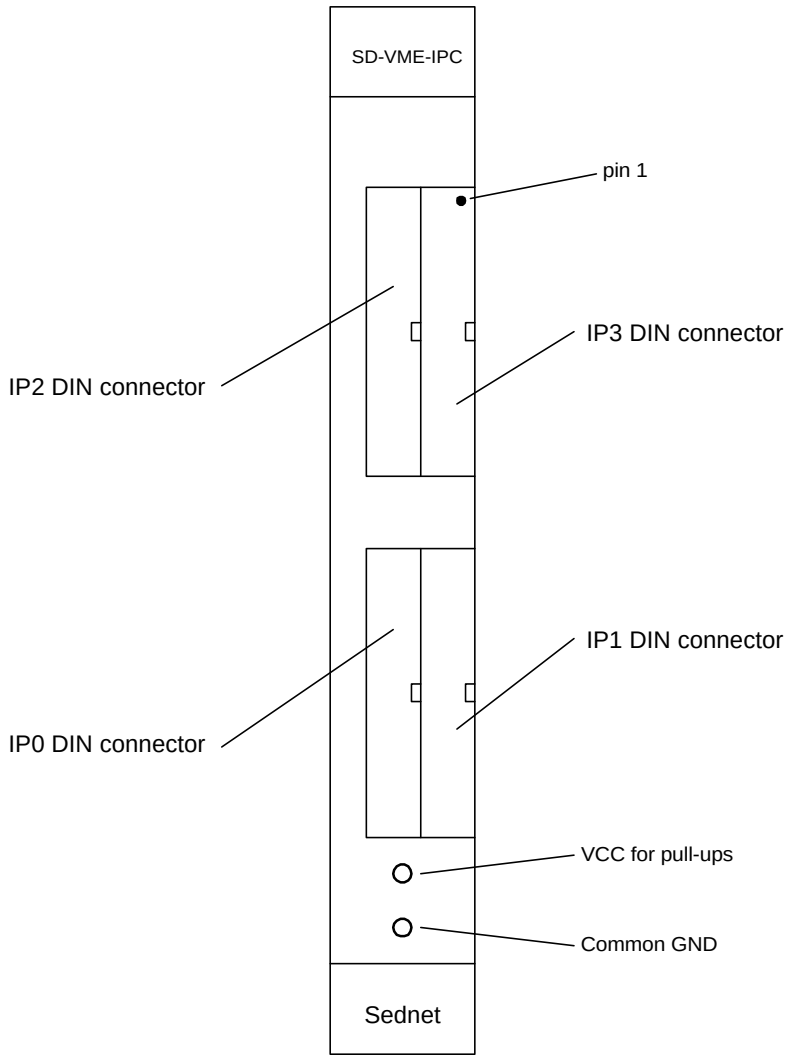


Figure 13 Front panel view of the SD-VME-IPC board

6.3.2 SD-VME-IPA Front Panel View

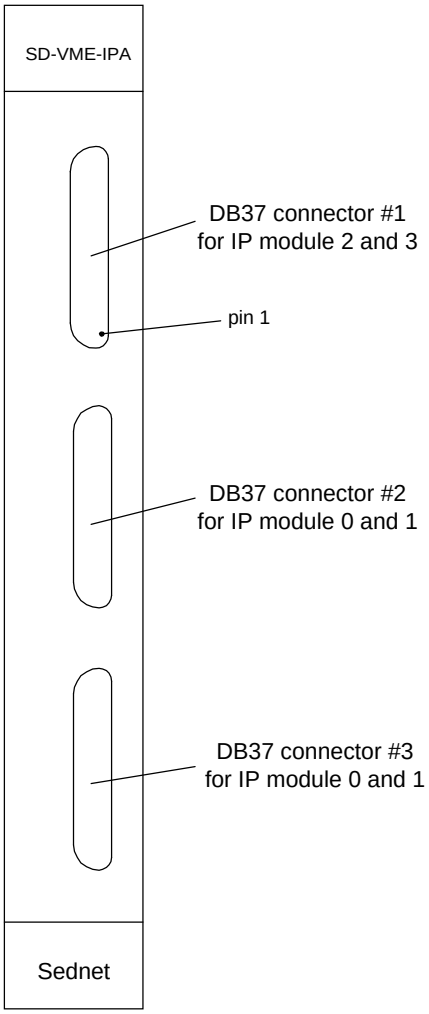


Figure 14 Front panel view of the SD-VME-IPA board

6. 4 Memory Mapping of the IP Modules

The following table is the memory mapping for the IP modules:

	IP module 0	IP module 1	IP module 2	IP module 3	
ID SEL (64 bytes)	0x09000000	0x09000080	0x09000100	0x09000180	IP carrier board #1
INT SEL (64 bytes)	0x09000040	0x090000C0	0x09000140	0x090001C0	
IO SEL (16 bytes)	0x09000400	0x09000500	0x09000600	0x09000700	
ID SEL (64 bytes)	0x09400000	0x09400080	0x09400100	0x09400180	IP carrier board #2
INT SEL (64 bytes)	0x09400040	0x094000C0	0x09400140	0x094001C0	
IO SEL (16 bytes)	0x09400400	0x09400500	0x09400600	0x09400700	
ID SEL (64 bytes)	0x09800000	0x09800080	0x09800100	0x09800180	IP carrier board #3
INT SEL (64 bytes)	0x09800040	0x098000C0	0x09800140	0x098001C0	
IO SEL (16 bytes)	0x09800400	0x09800500	0x09800600	0x09800700	

Table 16 Memory mapping for IP modules

The following table shows a memory mapping for the IO SEL that allows access to all the IP modules' IO SEL in a continuous memory space, therefore requiring less memory access:

	IP Module 0	IP Module 1	IP Module 2	IP Module 3	
IO SEL (16 bytes)	0x09000200	0x09000210	0x09000220	0x09000230	IP Carrier board #1
IO SEL (16 bytes)	0x09000240	0x09000250	0x09000260	0x09000270	IP Carrier board #2
IO SEL (16 bytes)	0x09000280	0x09000290	0x09000300	0x09000310	IP Carrier board #3

Table 17 Memory mapping for the IO SEL

[1] ID SEL is the location that contains specific information about your IP modules (Vendor ID, Product ID, IP type, etc. See your IP documentation for details.)

7. USER SPECIFIC FUNCTIONS (USF)

With the SD-VME-400 board interface, you can implement your own functions called *User Specific Functions* (USF), which are used to execute your own sequences of instructions. For example, a USF can be used to configure the VME communication adapter, transmit or receive asynchronous packets or update the communication adapter's LEDs or four-character display screen. You can create these functions using routines that are implemented in a library.

An example of an USF is provided in Section 7. 7. A selection of routines you can use to create the USF are described in Section 7. 6.

7. 1 Parameters

Two parameters are associated with USF.

Address location	Name	Definition	Default Value
0xb0008	USER_FUNCTION	0 = Idle 1 = Start 2 = Start if P10 is not connected 3 = Start if P10 is connected	0 = Idle
0xb000c	USER_FUNCTION_ADD	Address of the user function	0xc0000

The USER_FUNCTION parameter is used to start the USF and the USER_FUNCTION_ADD defines the start address of the USF. You can modify these parameters by writing different values to their addresses.

These parameters are set with default values read from the FLASH memory at start up. The corresponding parameters are USER_FUNCTION_FLAG and USER_FUNCTION_ADD.

Address location	Name	Definition	Default Value
0xAFF2C	USER_FUNCTION_FLAG	Flag for user functions	0 = Idle
0xAFF30	USER_FUNCTION_ADD	Address of the user function	0xc0000

Table 18 USF parameters

You can set and save these parameters with the EVM-400-SI software interface program.

7. 2 Creating a User Specific Function (USF)

To create a User Specific Function, `usf.c`, use the functions described in Section 7. 6. An example you can use as a guide is provided in Section 7. 7. The basic steps to follow are given below.

1. Select the destination addresses where the user specific function will be uploaded in memory, if you wish to modify the default values. The destination address for the function (ROM2) and the data (RAM1) must both be set. These addresses are specified in the `link.spc` and `usf.spc` files. Be sure that the locations you choose are in the user SRAM. See Sections 7. 8 and 7. 9 for examples of the `link.spc` and `usf.spc` files respectively.
2. Compile the `usf.c file` with the makefile `mr.bat`. This will create the files you need to create the S-Record file.
3. Execute the `usf.bat` file to create the S-Record file, `usf.dwn`.
4. Use the EVM-400-SI software to update the User Specific Function in the FLASH memory of the selected VME communication adapter. (See Section 7. 3 for details.)
5. Click the *Quit* button to return to the main screen of the EVM-400-SI program.

7. 3 Updating a User Specific Function (USF) in FLASH memory

Use the EVM-400-SI application to update the USF in the VME communication adapter's FLASH memory.

1. Choose *Update USF* from the *Update* pull-down menu. The *Update USF* dialogue box will appear.
2. Indicate the bus and node ID of the communication adapter whose firmware you wish to update in hexadecimal.
3. Select the *Motorola S-Record Format* file with the file extension *.DWN* containing the update to be written to the firmware of the selected VME communication adapter. When you click the *File name* textbox, a dialogue box will appear where you select the appropriate *DWN* file. The truncated path of the file you choose will be displayed in this textbox once the dialogue box closes.
4. Click the *Send* button to download the User Specific Function to the VME communication adapter you selected in the *Update USF* dialogue box.
5. Click the *Quit* button to return to the main screen of the EVM-400-SI program.

7. 4 Executing a User Specific Function (USF)

1. Update the VME communication adapter's FLASH using your USF.
2. Specify the USF's start address by setting the `USER_FUNCTION_ADD` parameter
3. Set the appropriate value of the `USER_FUNCTION` from Table 19:

Parameter Value	Action
1	Starts the USF
2	Starts the USF if jumper P10 is not connected
3	Starts the USF if the jumper P10 is connected

Table 19 USER_FUNCTION parameter values

If you need to run the USF at startup, update the `USER_FUNCTION_FLAG` and `USER_FUNCTION_ADD` parameters in the FLASH memory with the appropriate values.

Caution: If you set the USER_FUNCTION_FLAG to 1, the USF will be run at each startup.

7. 5 Available USF Structures

7.5.1 p_1394_packet_t

SYNOPSIS

```
typedef struct _p_1394_packet_s {
      unsigned long   add_packet;      /* Address of packet */
      unsigned long   add_data;        /* Address of data */
      unsigned long   length;          /* Length in quadlets */
} p_1394_packet_t;
```

DESCRIPTION Defines the address and length of the packet to be transmitted or received.

PARAMETERS *add_packet* – The address of packet to be transmitted or received

add_data - The address of the packet's data

length - The length of data contained in the packet, expressed in quadlets.

7.5.2 sdNET_Packet_t

SYNOPSIS

```
typedef struct sdNET_Packet_s {

      ULONG             tCode;
      union {
            struct {
                  ULONG bus;
                  ULONG node;
                  ULONG rCode;
                  ULONG offsetHigh;
                  ULONG offsetLow;
                  ULONG xtCode;
                  ULONG tLabel;
                  ULONG retryCode;
                  ULONG priority;
            } asy;

            struct {
                  ULONG channel;
                  ULONG tag;
                  ULONG synchro;
            } iso;
      } type;
      ULONG datalen;
      ULONG ack;
      ULONG speed;
      PULONG      data;
} sdNET_Packet_t, *PsdNET_Packet_t;
```

DESCRIPTION Defines all fields of an IEEE-1394 asynchronous packet. These fields are used to format a packet to be transmitted or read the information in a received packet.

PARAMETERS *tcode* – The code indicating the transaction type. See *IEEE 1394-1995 6.2.4.5*

bus – The bus ID of the incoming or outgoing packet. See *IEEE-1394-1995 6.2.4.2.1*

node – The node ID of the incoming or outgoing packet. See *IEEE-1394-1995 6.2.4.2.1*

rCode – The response code for incoming response packets. See *IEEE-1394-1995 6.2.4.10*

offsetHigh – The 16 bits high address. See *IEEE-1394-1995 6.2.4.2.2*

offsetLow – The 32 bits low address. See *IEEE-1394-1995 6.2.4.2.2*

xtCode – The extended transaction code. See *IEEE-1394-1995 6.2.4.9*

tLabel – The label uniquely identifying each transaction. See *IEEE-1394-1995 6.2.4.3*

retryCode – The code indicating whether or not this packet is a retry. See *IEEE-1394-1995 6.2.4.4*

priority – The priority for the PHY backplane. See *IEEE-1394-1995 6.2.4.6*

channel – The packet's isochronous channel number. See *IEEE-1394-1995 6.2.4.13*

tag – The isochronous packet's tag. See *IEEE-1394-1995 6.2.4.12*

synchro – The synchronization code for isochronous transmission. See *IEEE-1394-1995 6.2.4.14*

datalen – The packet's data length. See *IEEE-1394-1995 6.2.4.8*

ack – The acknowledge code for asynchronous packets. See *IEEE-1394-1995 6.2.5.2.2*

speed – The speed at which the packet was sent. 00 = 100 Mbps, 01 = 200 Mbps, 10 = 400 Mbps, and 11 is undefined.

data – The information to be transmitted in quadlet format. See *IEEE-1394-1995 6.2.4.11*

7. 6 Available USF Functions

7.6.1 SD_Reset_1394

SYNOPSIS

```
int     SD_Reset_1394()
```

DESCRIPTION Resets the 1394 physical and logical interface hardware (bus reset).

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful.

7.6.2 SD_Check_port_1394

SYNOPSIS

```
int  SD_Check_port_1394()
```

DESCRIPTION Verify which 1394 ports, if any, are connected to the serial bus.

INPUT None

OUTPUT None

RETURNS

`0` - No port connected

`1` – Port #1 connected

`2` – Port #2 connected

`3` – Ports #1 and #2 connected

`4` – Port #3 connected

`5` - Ports #1 and #3 connected

`6` – Ports #2 and #3 connected

`7` – Ports #1, #2 and #3 connected

`8` - `ADAPTER_NOT_FOUND`. The required adapter ID was not found

7.6.3 SD_Init_Adapter_1394

SYNOPSIS

```
int     SD_Init_adapter_1394()
```

DESCRIPTION Initializes the 1394 Logical Interface Register. When you call this service, the Control register is set to `0x86700001`, the Interruption Mask register is set to `0x80000000` and FIFO Control register is set to `0x00020000`.

INPUT None

OUTPUT None

RETURNS

1 – SUCCESS. The call was successful
8 - ADAPTER_NOT_FOUND. The required adapter ID was not found

7.6.4 SD_Receive_1394

SYNOPSIS

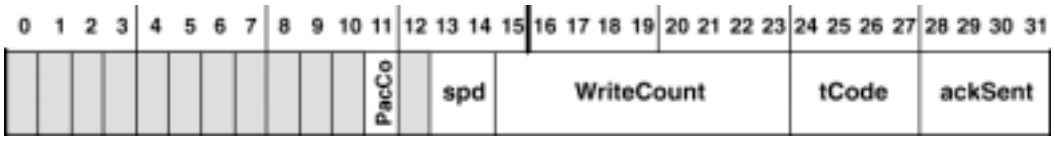
```
int SD_Receive_1394(
            p_1394_packet_t *p_packet)
```

DESCRIPTION Receives an IEEE-1394 asynchronous packet from the GRF FIFO of the 1394 Logical Interface FIFO.

INPUT *packet – Pointer to a p_1394_packet_t structure that defines the IEEE-1394 asynchronous packet to be received.

OUTPUT None

RETURNS -12 – DATA_NOT_PRESENT. No data is present
Other – Extra-header (See *TSB12LV01A Data Manual*, Sections 4.1.3 and 4.1.4):



Bit 11 = PacCo
Bits 13-14 = Speed
Bits 15 to 23 = WriteCount
Bits 24 to 27 = tCode
Bits 28 to 31 = ackSent

Figure 15 1394 packet header bit values

7.6.5 SD_Send_asy_1394

SYNOPSIS

```
int SD_Send_asy_1394(
            p_1394_packet_t *p_packet)
```

DESCRIPTION Sends an IEEE-1394 asynchronous packet to the ATF FIFO of the 1394 Logical Interface FIFO.

INPUT `*packet` – Pointer to the `p_1394_packet_t` structure that defines the IEEE-1394 asynchronous packet to be sent.

OUTPUT None

RETURNS

`0` – `FALSE`. The call failed
`1` – `SUCCESS`. The call was successful

7.6.6 SD_mkpacket_1394

SYNOPSIS

```
int     SD_mkpacket_1394(
                sdNET_Packet_t *pk,
                p_1394_packet_t *p_packet)
```

DESCRIPTION Creates a pre-formatted IEEE-1394 asynchronous packet with the parameters defined at `*pk`.

INPUT `*pk` – Pointer to the `sdNET_Packet_t` structure which defines all fields of an IEEE-1394 asynchronous packet.

OUTPUT `*p_packet` – Pointer to `p_1394_packet_t` structure. This structure will be filled after the IEEE-1394 asynchronous packet is created.

RETURNS

`0` – `FALSE`. The call failed
`1` – `SUCCESS`. The call was successful

7.6.7 SD_Init_IDMA

SYNOPSIS

```
int SD_Init_IDMA()
```

DESCRIPTION Configures IDMA channels 1 and 2. More specifically, this function sets the IDMA channel configuration register. The IDMA channels are set to ignore the FREEZE signal, the IDMA channel 1 is given priority over IDMA 2, the interrupt service masks are enabled, and the arbitration ID is set to 6.

INPUT None

OUTPUT None

RETURNS
`1` – SUCCESS. The call was successful

7.6.8 SD_LED_Error_on

SYNOPSIS
```
int SD_LED_Error_on()
```

DESCRIPTION Lights up the Error LED.

INPUT None

OUTPUT None

RETURNS
`1` – SUCCESS. The call was successful

7.6.9 SD_LED_Error_off

SYNOPSIS
```
int SD_LED_Error_off()
```

DESCRIPTION Switches off the Error LED.

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.10 SD_LED_Run_on

SYNOPSIS
```
int SD_LED_Run_on()
```

DESCRIPTION Lights up the Run LED.

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.11 SD_LED_Run_off

SYNOPSIS

```
int SD_LED_Run_off()
```

DESCRIPTION Switches off the Run LED.

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.12 SD_LED_Comm_on

SYNOPSIS

```
int SD_LED_Comm_on()
```

DESCRIPTION Lights up the Communication LED.

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.13 SD_LED_Comm_off

SYNOPSIS

```
int SD_LED_Comm_off()
```

DESCRIPTION Switches off the Communication LED.

INPUT None

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.14 SD_Init_digit

SYNOPSIS

```
int SD_Init_digit(int digit_on)
```

DESCRIPTION Switches an LCD digit ON or OFF.

INPUT `digit_on` – 0 = Off, 1 = On

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7.6.15 SD_Update_digit

SYNOPSIS

```
int SD_Update_digit(unsigned char *ptr_c)
```

DESCRIPTION Updates the four-character LCD display.

INPUT `*ptr_c` – Pointer to an array of four characters.

OUTPUT None

RETURNS `1` – SUCCESS. The call was successful

7. 7 USF Example

In this example, the 1394 bus is initialized, reception of 1394 packets is initialized, and the device sends and is readied to receive an asynchronous packets.

```
/*****************************************************************/
/* File name :          usf.c                                    */
/* Descript. :          USF Example                              */
/* Version   :          1.02                                     */
/* Date      :          03-22-99                                 */
/* Author    :          C. Brayet                                */
/* Company   :          Sederta, Product Division of Mindready Inc.  */
/*****************************************************************
```

```
*********************************************************************
File description :

        void main()

*********************************************************************/

#include "p1394.h"
#include "tsb12c01.h"
#include "sdfwi.h"
#include "io.h"
#include "ieee1394.h"

/* Global variables */

/* External functions */

/*********************************************************************/


/*********************************************************************/
/*  Main functions:                                                  */
/*              - initialize the 1394                                */
/*              - send asynchronous 1394 pre-format packet           */
/*              - wait for an asynchronous packet                    */
/*********************************************************************/

void main()
{
int                 i,j;
int                 err, port;
unsigned char       ch[4];

sdNET_Packet_t      out_1394_pk;     /* 1394 structure definition */
                                     /* (sdfwi.h) */
p_1394_packet_t     p_out_pk;        /* 1394 sent packet address */
                                     /* Definition (sdfwi.h) */
unsigned long       out_pk[MAX_DATA_QUADLET];    /* 1394 packet */
p_1394_packet_t     p_in_pk;         /* 1394 received packet address */
                                     /* Definition (sdfwi.h) */
unsigned long       in_pk[MAX_DATA_QUADLET];     /* 1394 packet */
unsigned long       *ptr_out;
unsigned long       *ptr_in;


      /* Initialization, without interrupt */
      if((err = SD_Init_adapter_1394()) != SUCCESS)
                SD_LED_Error_on();

      /* Update LCD digits */
      ch[0]='H'; ch[1]='Y'; ch[2]='P'; ch[3]='R';

      port = SD_Check_port_1394();
      if(port & 4)      ch[1]= ch[1] + 43;
      if(port & 2)      ch[2]= ch[2] + 43;
      if(port & 1)      ch[3]= ch[3] + 43;
      if(port == ADAPTER_NOT_FOUND) {
               ch[0]='E'; ch[1]='R'; ch[2]='R'; ch[3]='6'; }

      err = SD_Update_digit(&ch[0]);
```

```
	/* Set the LED RUN */
	err = SD_LED_Run_on();

	/* Initialize 1394 packet reception */
	p_in_pk.add_packet = (unsigned long)&in_pk[0];

	/* Definition of the 1394 asynchronous packet (ieee1394.h) */
	out_1394_pk.type.asy.bus = 0;
	out_1394_pk.type.asy.node = 0xb;
	out_1394_pk.type.asy.offsetLow = 0xc0000;
	out_1394_pk.type.asy.offsetHigh = 0;
	out_1394_pk.type.asy.tLabel = 0;
	out_1394_pk.speed = IEEE1394_ASYSPD200Mbits;
	out_1394_pk.type.asy.retryCode = IEEE1394_Retry_0;
	out_1394_pk.tCode = IEEE1394_BLOCK_WRREQ;
	out_1394_pk.datalen = 32;	/* 8 quadlets */
	p_out_pk.add_packet = (unsigned long)&out_pk[0];

	err = SD_mkpacket_1394(&out_1394_pk, &p_out_pk);


	/* Data in packet */
	ptr_out = (unsigned long *)p_out_pk.add_data;
	ptr_in = (unsigned long *)p_in_pk.add_data;
	for(i=0; i<8; i++) {
		*ptr_out = i + 0x12340000;
		ptr_out++;
	}

	/* Send asynchronous 1394 pre-formatted packet */
	err = SD_Send_asy_1394(&p_out_pk);

	/* Wait for an asynchronous packet */
	while((err = SD_Receive_1394(&p_in_pk)) == DATA_NOT_PRESENT)
	{
		for (i=0; i<2; i++){
			SD_LED_Run_on();
			for (j=0; j<100000; j++);
			SD_LED_Run_off();
			for (j=0; j<100000; j++);}
		for (j=0; j<250000; j++);
	}

	for (j=0; j<5000000; j++);

	/* Set the LED COMM */
	err = SD_LED_Comm_on();

}
```

7. 8 usf.spc file example

```
/*********************************************************************
   Example specification file for the 68000 family.  This example
```

```
   assumes ROM is at address 0x0000 and RAM is at address 0xC0000.
   Region "data" (which contains initialized RAM variables) must
   be linked into RAM to give the variables their correct addresses,
   but will be downloaded into ROM at location DATA using downloader
   option "-m data,DATA".  Startup code will copy it from location
   DATA back into RAM.
*********************************************************************/

partition { overlay {
    CODE_PROG = $;
    region {} code;                     /* Executable code */
    region {} string;                   /* Constant strings */
    region {} const;                    /* Constant data */
} o2; } ROM2[addr=0xC0000];             /* Set the address here */

partition { overlay {
    DATA = $;
    region {} data[roundsize=4];        /* RAM to be initialized on reset */
    region {} ram[roundsize=4];         /* RAM to be zeroed on reset */
    region {} buffer;                   /* Executable code */
    region {} malloc[size=0x1000];      /* RAM available to malloc() */
} o3; } RAM1[addr=0xE0000];             /* Set the address here */
```

7. 9 Link.spc file example

```
/*********************************************************************
   User Specific Function (USF)
   Example specification file for the 68000 family.  This example
   assumes RAM is at address 0xC0000.
*********************************************************************/
partition { overlay {
    region {} code;                     /* executable code */
    region {} const;                    /* constant data */
    region {} string;                   /* constant strings */
} o2; } RAM1[addr=0xC0000];             /* Set the address here */

partition { overlay {
    DATA = $;
    region {} data[roundsize=4];        /* RAM to be initialized on reset */
    region {} ram[roundsize=4];         /* RAM to be zeroed on reset */
    region {} malloc[size=0x1000];      /* RAM available to malloc() */
} o3; } RAM0[addr=0xE0000];             /* Set the address here */
```

8. OTHER FEATURES

8. 1 Writing your Own Parameters to the User FLASH Memory

You can write your own parameters and values to the FLASH memory. To do so, follow the procedure below:

Action	Address	Value
1. Write your data in a memory zone other than the FLASH area. (The data will be copied from this zone to the FLASH.)	Address of the memory zone where the data is located	Data Value
2. Write the address offset low of the data block to copy to the FLASH memory in `0x000AFFC4`. (The same address as above.)	`0x000AFFC4`	Address of the data block to copy
3. Write the length in bytes of the data block to copy to the FLASH memory in the 16 most significant bits at address `0x000AFFC0`.	`0x000AFFC0`	|32 length 16|15 origin 0|
4. Write the address offset high of the data in the 16 least significant bits of address `0x000AFFC0`. (Write a 0 if on the card, or `0x1A` if on the VME bus.)	`0x000AFFC0`	|32 length 16|15 origin 0|
5. Write the address of the FLASH memory block where the data will be copied in `0x000AFFC8`.	`0x000AFFC8`	Address of the FLASH memory block
6. Start the Update FLASH semaphore by writing a 1.	`0x000AFFBC`	`0x000000001`

Table 20 Writing user-defined parameters and values to FLASH memory

When updating the FLASH, the system should display the update's percentage progress and return to bus/node display when the update is over.

Caution: Performing a FLASH update may cause loss of all data in the user SRAM because of the card reboot process.

8. 2 Detecting the Number of 1394 Adapters Connected to the 1394 Bus with Self-ID Packets

By reading at address `0x000AFD00`, you can read a FIFO with a certain amount of self-ID packets. These packets tell the VME-400 card how many adapters are connected to the bus and what is the exact configuration (topology) of the 1394 bus.

These packets are 64-bit and can be interpreted by reading Section 4.3.4.1 of the IEEE-1394 (1995) Standard. (*IEEE Standard for a High Performance Serial Bus*)

They are written in a sequential fashion in the memory space starting at `0x000AFD00` and can be read until a 32-bit word (quadlet) containing `0x00000000` is read, which means that no more self-ID packets can be found in the FIFO.

ANNEX I: SRAM PARAMETERS

Parameter	Description	Address Offset
IEEE_1394_VENDOR_ID	The 1394 vendor ID identifies the board manufacturer. Default is 0x30DB	0x000AFEDC
CHIP_ID_HIGH	This is the high part of the chip identifier, which is part of the Node Unique ID.	0x000AFEE0
CHIP_ID_LOW	This is the low part of the chip identifier, which is part of the Node Unique ID.	0x000AFEE4
IEEE_1394_COMPLIANCE	1 -> 1394 Compliant 0 -> SedNet Compliant	0x000AFEE8
MAX_REC	This is the maximum size of a received packet on the 1394 bus. It is calculated with this formula: packet size = 2^{MAX_REC+1}	0x000AFEEC
ISOCH_ENABLE	This option enables isochronous transactions.	0x000AFEF0
BUS_MANAGER_ENABLE	This option enables bus manager capabilities. (currently unavailable)	0x000AFEF4
ISOCH_MANAGER_ENABLE	This option enables isochronous manager capabilities. (currently unavailable)	0x000AFEF8
CYCLE_MASTER_ENABLE	This option enables cycle master capabilities.	0x000AFEFC
SERIAL_NUMBER	The serial number of the card	0x000AFF00
VERSION_NUMBER	The version number of the software on the card	0x000AFF04
BUS_NODE	1394 node and bus number	0x000AFF08
RESPONSE_FLAG	1394 response flag (a response is sent if the flag is 0x1)	0x000AFF0C
ISO_REC_CHAN_1_OFF	Isochronous1 channel number	0x000AFF10
ISO_REC_CHAN_2_OFF	Isochronous2 channel number	0x000AFF14
CS5_ASIZ	Address size and data size of the VME	0x000AFF18
SLAV_ADD	Base address of the memory map that can be accessed on the card	0x000AFF1C
SLAV_ADD_MSK	Mask for the address on the card	0x000AFF20
VME_ADD	Address of the card on the VME bus	0x000AFF24
VME_ADD_MSK	Mask for the address on the VME bus	0x000AFF28
USER_FUNCTION_FLG	Flag for user function.	0x000AFF2C
USER_FUNCTION_ADD	Address of user function.	0x000AFF30
ISO_REC_ENABLE_OFF	Enables the receiver channel 0 = All channels disabled 1 = Channel 1 enabled 2 = Channel 2 enabled 3 = Both channels are enabled	0x000AFF34
ISO_REC_BLOCK_LENGTH_1_OFF	Length of the block of the memory to receive in byte (channel 1)	0x000AFF38
ISO_REC_FRAME_LENGTH_1_OFF	Length of the frame of the memory to receive in byte (channel 1)	0x000AFF3C

ISO_REC_DEST_HIGH_1_OFF	Address high of the block of the memory to write (channel 1)	0x000AFF40
ISO_REC_DEST_LOW_1_OFF	Address low of the block of the memory to write (channel 1)	0x000AFF44
ISO_REC_PTR_1_OFF	PTR of the address after a DMA transfer. Used to know where the pointer of the receive is (channel 1)	0x000AFF48
ISO_REC_SYNC_1_OFF	syMode: type of synchronization (bite 4 to 7) 0 = no synchronization 1 = synchronization at each corresponding syMode sy: transaction layer-specific synchronization bits. 0 = no synchronization (bits 0-3).	0x000AFF4C
ISO_REC_DONE_1_OFF	Reception of block has been completed when this bit is set to 1.	0x000AFF50
ISO_REC_BLOCK_LENGTH_2_OFF	Length of the block of memory to receive, expressed in bytes (channel 2).	0x000AFF54
ISO_REC_BLOCK_FRAME_2_OFF	Length of the frame of memory to be received, expressed in bytes (channel 2).	0x000AFF58
ISO_REC_DEST_HIGH_2_OFF	Address high of the block of the memory to be written (channel 2).	0x000AFF5C
ISO_REC_DEST_LOW_2_OFF	Address low of the block of the memory to be written (channel 2).	0x000AFF60
ISO_REC_PTR_2_OFF	PTR of the address after a DMA transfer. Used to know where the pointer of the receiver is (channel 2)	0x000AFF64
ISO_REC_SYNC_2_OFF	sy mode: type of synchronization (bits 4-7) sy: transaction layer-specific synchronization bits. 0 = no synchronization (bit 0-3)	0x000AFF68
ISO_REC_DONE_2_OFF	Reception of block has been completed when this bit is set to 1.	0x000AFF6C

ANNEX II: IMPLEMENTED CSRs

The following Control and Status Registers (CSRs) have been fully or partially implemented for this version of the VME-400 communication adapter.

Refer to the *IEEE Standard for a High Performance Serial Bus (IEEE Std 1394-1995)* or the *ANSI/IEEE Std 1212-1994* for further details concerning the how and why of these CSRs.

CSR	Address Offset	Access Type	Reset Value
State Clear	FFFF F000 0000	R/W	0x00000000
State Set	FFFF F000 0004	WO	-
Node ID	FFFF F000 0008	R/W	0xFFC00000
Reset Start	FFFF F000 000C	WO	-
Split Timeout Hi	FFFF F000 0018	R/W	0x00000000
Split Timeout Low	FFFF F000 001C	R/W	0x80000000
Busy Timeout	FFFF F000 0210	R/W	0x00200000
Bus Info	FFFF F000 0400	RO	-
Bus Info: Bus Type	FFFF F000 0404	RO	0x31333934
Bus Info: capability	FFFF F000 0408	RO	0x00FF8000
Bus Info: vendor ID Chip ID Hi	FFFF F000 040C	RO	0x0030DB00
Bus Info: Chip ID Low	FFFF F000 0410	RO	-

ANNEX III: SEMAPHORES

A *semaphore* is a message sent to the application telling that an action needs to be performed or has been performed. When a value is written at a certain address, the CPU will know that it must start some actions. Once the action is completed, the CPU usually answers back by telling that the action is done.

This is a table with all the semaphores mapped on the card. Most of the time, writing a `0x01` to the address will start the semaphore and a `0x00` will be written when the task is completed.

Address	Semaphore	Description
`0x000AFF80`	RESET	This semaphore causes a software reset of the board being reset.
`0x000AFF88`	Save parameters in FLASH	This semaphore takes the parameters in the SRAM and writes them to the user FLASH. This is the only way to change the parameters in the FLASH since it cannot be accessed by the user directly. The values in the FLASH are the ones given to the system at boot-up.
`0x000AFF8C`	Restore default parameters	This semaphore takes the parameters in the default FLASH section and copies them to the SRAM. This is to be used only when you wish to restore the system to its default values.
`0x000AFF90`	Update parameters	This semaphore takes the parameters in the user FLASH area and copies them to the SRAM.
`0x000AFF94`	Bus Error	This semaphore is used internally to monitor bus errors. (Reserved for use by the system)
`0x000AFFA8`	1394 bridge	This semaphore is used as a request for 1394 packet transmission when the card is used as a bridge for 1394 packets. It also monitors the state of the 1394 devices.

ANNEX IV: MEMORY MAPPING OF THE BOARD

This is the general memory mapping of the card. It contains all memory locations available from the card:

Address	Memory or I/O	Description		Permissions
0x00000000 to 0x0003FFFF	Flash Memory (reserved for firmware)	256K of reserved FLASH space (512K, 8-bit)		Read Only
0x00040000 to 0x0007FFFF	Flash Memory	256K of user FLASH space (512K, 8-bit)		R/W
0x00080000 to 0x000BFFFF	Static Memory Bank 0	Static RAM (256K, 32-bit) (program SRAM area)		Read Only
0x000C0000 to 0x000FFFFF	Static Memory Bank 1	Static RAM (256K, 32-bit) (user SRAM area)		R/W
0x00100000	1394 Link Layer (TSB12C01)	This controller transmits and receives correctly-formatted 1394 packet		Read Only
0x00100100	1394 state machine			
0x01000000 to 0x01FFFFFF	DRAM SIMM 16M – Bank 0	72-PIN SIMM DRAM (16M – 32-bit)		R/W
0x02000000 to 0x02FFFFFF	DRAM SIMM 16 M – Bank 1	72-PIN SIMM DRAM (16M – 32 bit) (32M option only)		
0x09000400	IP MODULE 0	First IP module	First IP Carrier Board	
0x09000500	IP MODULE 1	Second IP module		
0x09000600	IP MODULE 2	Third IP module		
0x09000700	IP MODULE 3	Fourth IP module		
0x09400400	IP MODULE 4	First IP module	Second IP Carrier Board	
0x09400500	IP MODULE 5	Second IP module		
0x09400600	IP MODULE 6	Third IP module		
0x09400700	IP MODULE 7	Fourth IP module		
0x09800400	IP MODULE 8	First IP module	Third IP Carrier Board	
0x09800500	IP MODULE 9	Second IP module		
0x09800600	IP MODULE 10	Third IP module		
0x09800700	IP MODULE 11	Fourth IP module		

This is the memory mapping for the 1394 link layer that transmits and receives 1394 packets. It describes the TSB12LV01 registers:

Address	Register	Description
0x00100000	Version	Version and Revision register
0x00100004	Node Address	Node address and Transmitter acknowledge
0x00100008	Control	Control register
0x0010000C	Interrupt	Interrupt register
0x00100010	Interrupt Mask	Interrupt Mask register
0x00100014	Cycle Timer	Cycle-timer register
0x00100018	Isoch. Port Number	Isochronous Receive Port number register
0x0010001C	FIFO Control	Controls FIFO allocation or clears the FIFO's contents
0x00100020	Diagnostics	Diagnostic Control and Status register
0x00100024	Phy Chip Access	Access register for the physical chip
0x00100028	Reserved	
0x0010002C	Reserved	
0x00100030	ATF Status	Asynchronous Transmit-FIFO Status register
0x00100034	ITF Status	Isochronous Transmit-FIFO Status register
0x00100038	Reserved	
0x0010003C	GRF Status	General Transmit-FIFO Status register (read-only)
0x00100040	Reserved	

ANNEX V: ERROR MESSAGES

Error messages may be displayed on your LCD display to let you know that an error was encountered on your system.

Some errors may be minor; whereas others may be significant or even unrecoverable.

The list below summarizes the most common errors that could happen on your system:

Error Message	Description
ERR1	SRAM error
ERR2	DRAM error
ERR3	1394 initialization error
ERR6	No 1394 adapter found
ERR7	Flash Update error
ERR8	Bus error
ERR9	1394 operation timeout
ERR10	VME operation timeout
ERR11	Reserved
ERR12	Reserved
ERR13	Reserved
ERR14	Reserved
ERR15	Reserved
ERR16	Reserved